MIS-100

Title of the Invention

COMPUTER DATABASE SYSTEM AND METHOD FOR COLLECTING AND REPORTING REAL ESTATE PROPERTY AND LOAN PERFORMANCE INFORMATION OVER A COMPUTER DRIVEN NETWORK

Inventor

Shaun Michael Brady

COMPUTER DATABASE SYSTEM AND METHOD FOR COLLECTING AND REPORTING REAL ESTATE PROPERTY AND LOAN PERFORMANCE INFORMATION OVER A COMPUTER DRIVEN NETWORK

Field of the Invention

The invention relates to a computer database system and method using a database in a computer database system for collecting and processing information from data providers of real estate property and loan performance information and providing reports concerning the information in response to queries, and in particular to collecting and reporting the property and loan performance information over a computer driven network.

Background of the Invention

Based on numerous public and private studies, it is widely agreed that the real estate data market lacks a consistent, standardized, and timely centralized source of property and loan performance information, especially with respect to the multifamily housing industry. Consequently, the economic cycles affecting real estate are more severe than they would be otherwise, costing the nation billions of dollars in lost revenues, productivity, and affordable housing stock.

Many of the industry's larger owners, managers, financial institutions, and rating and research organizations have long felt the need to standardize and share their confidential and proprietary performance data under an agreed set of 'confidentiality principals'. However, no such agreement has existed to govern the way in which data is collected, secured, and in particular processed into statistical reports that can

be made publicly available. A main concern with respect to such an agreement has involved outlining a reporting process that does not reveal the confidential aspects of the underlying assets and related properties.

Data of the type needed for building a database of real estate property and loan performance information has typically been maintained by an independent research company or organization. Such a company or organization maintains the confidentiality of the data by checking each report request and following strict guidelines governing the confidentiality of the data. However, data providers are concerned that the same measures of confidentiality cannot be maintained when the report queries are processed automatically and responses forwarded electronically over a computer driven network, such as the Internet.

Accordingly, despite the availability of the Internet and database servers, in general, the real estate data market is still under served by this technology. The primary reason for this is the reluctance of the asset managers and property owners, who have the proprietary data needed for building an adequate database, to come forward and provide the data. The reluctance is due to their concern for maintaining the confidentiality of the data. That is, if confidential data of assets is entered into a database that is available for searching over the Internet, then the risk for the data provider is that the data may be made available to the public, at great harm to the individual data provider.

Summary of the Invention

It is an object of the invention to collect, store and provide reports on real estate property and loan performance information provided by data providers from the real estate industry, especially the multifamily real estate industry,

using a computer database and a computer driven network, such as the Internet.

It is an object of the invention to establish a computer database into which data of real estate assets, including detailed information identifying the property and its usage, as well as loan information and financial performance information is entered into the database by data providers in a standardized format over a computer driven network.

Further, it is an object of the invention to generate reports from the information stored in the database, including pre-aggregated and user defined reports without the need for manual processing of the report requests that are forwarded to users over a computer driven network without compromising the confidentiality of the data of any of the underlying assets.

Brief Description of the Drawings

Fig. 1 shows the network architecture for the system of the present invention.

Fig. 2 shows an overview of the data flow in the system of Fig. 1.

Fig. 3 shows a user interface for selecting geographic area or market information in a user query process.

Fig. 4 shows a user interface for entering property information in a user query process.

Fig. 5 shows a user interface for entering time period information in a user query process.

Fig. 6 shows a user interface for entering financing information in a user query process.

Fig. 7 shows a user interface for entering report format details in a user query process.

Fig. 8 is a table showing the description of data elements stored in the database of the system.

Fig. 9 is a table of loan static information.

Fig. 10 is a table of additional loan static information.

Fig. 11 is a table of loan dynamic information.

Fig. 12 is a table of property static information.

Fig. 13 is a table of additional property static information.

Fig. 14 is a table of additional property static information.

Fig. 15 is a table of property dynamic performance information.

Fig. 16 is a table of additional property dynamic performance information.

Fig. 17 is a table of property valuation information.

Fig. 18 is a table of affordable housing information.

Fig. 19 a table of data structures used in explaining the data load process according to the present invention.

Fig. 20 is a table showing an example of the standard text file format used for the control file (header) in providing data to be stored in the database of the system.

Fig. 21 is a data flow diagram showing data submission of data provided by data providers according to the present invention.

Fig. 22 is a diagram of the call flow and corresponding tables used in the data submission process shown in Fig. 22.

Fig. 23 is a data flow diagram showing data validation of data provided by data providers according to the present invention.

Fig. 24 is a diagram of the call flow and corresponding tables used in the data validation process shown in Fig. 23.

Fig. 25 is a table of validation rules for loan static information.

Fig. 26 is a table of validation rules for loan static information.

Fig. 27 is a table of validation rules for loan dynamic information.

Fig. 28 is a table of validation rules for loan adjustable rate information.

Fig. 29 is a table of validation rules for loan prepayment information.

Fig. 30 is a table of validation rules for foreclosure or workout information.

Fig. 31 is a table of validation rules for property static information.

Fig. 32 is a table of validation rules for property static information.

Fig. 33 is a table of validation rules for property static information.

Fig. 34 is a table of validation rules for property dynamic information.

Fig. 35 is a table of validation rules for property dynamic information.

Fig. 36 is a table of validation rules for property dynamic information.

Fig. 37 is a table of validation rules for target rent information.

Fig. 38 is a table of validation rules for affordable housing program information.

Fig. 39 is a table of validation rules for property valuation information.

Fig. 40 is a table of validation rules for low-income housing tax credit information.

Fig. 41 is flow diagram showing the process of error review of data provided by data providers.

Fig. 42 is a data flow diagram showing asset matching of data provided by data providers with data already stored in the database according to the present invention.

Fig. 43 is a diagram of the call flow and corresponding tables used in the before matching sub-process as part of the asset matching process.

Fig. 44 is a diagram of the call flow and corresponding tables used in the matching sub-process as part of the asset matching process.

Fig. 45 a diagram of the call flow and corresponding tables used in the after matching sub-process as part of the asset matching process.

Fig. 46 is a diagram of the call flow and corresponding tables used in the migration process.

Fig. 47 is a data flow diagram showing data administration functions performed according to the present invention.

Fig. 48 a data flow diagram showing the user query process performed according to the present invention.

Fig. 49 a diagram of the call flow and corresponding tables used in the reporting process.

Fig. 50 is a diagram of a sample summary report generated by the system of the present invention.

Fig. 51 is a diagram of a sample detail report generated by the system of the present invention.

Fig. 52 is a diagram of a detail report statistics sample generated by the system of the present invention.

Fig. 53 is a table of the details of the control (header) file.

Fig. 54a is a table of the details of the loan static information data elements.

Fig. 54b is a table of additional details of the loan static information data elements.

Fig. 54c is a table of additional details of the loan static information data elements.

Fig. 54d is a table of additional details of the loan static information data elements.

Fig. 54e is a table of additional details of the loan static information data elements.

Fig. 55a is a table of the details of the loan dynamic information data elements.

Fig. 55b is a table of additional details of the loan dynamic information data elements.

Fig. 56a is a table of the details of the loan adjustable rate information data elements.

Fig. 56b is a table of additional details of the loan adjustable rate information data elements.

Fig. 57 is a table of the details of the loan loan prepayment information data elements.

Fig. 58a is a table of the details of the foreclosure or workout information data elements.

Fig. 58b is a table of additional details of the foreclosure or workout information data elements.

Fig. 59a is a table of the details of the property static information data elements.

Fig. 59b is a table of additional details of the property static information data elements.

Fig. 59c is a table of additional details of the property static information data elements.

Fig. 59d is a table of additional details of the property static information data elements.

Fig. 59e is a table of additional details of the property static information data elements.

Fig. 59f is a table of additional details of the property static information data elements.

Fig. 59g is a table of additional details of the property static information data elements.

Fig. 59h is a table of additional details of the property static information data elements.

Fig. 60a is a table of the details of the property dynamic performance information data elements.

Fig. 60b is a table of additional details of the property dynamic performance information data elements.

Fig. 60c is a table of additional details of the property dynamic performance information data elements.

Fig. 60d is a table of additional details of the property dynamic performance information data elements.

Fig. 60e is a table of additional details of the property dynamic performance information data elements.

Fig. 60f is a table of additional details of the property dynamic performance information data elements.

Fig. 60g is a table of additional details of the property dynamic performance information data elements.

Fig. 61a is a table of the details of the property valuation information data elements.

Fig. 61b is a table of additional details of the property valuation information data elements.

Fig. 62 is a table of the details of the low-income housing tax credit information data elements.

Fig. 63 is a table of the details of the target rent information data elements.

Fig. 64a is a table of the details of the affordable housing program information data elements.

Fig. 64b is a table of additional details of the affordable housing program information data elements.

Fig. 64c is a table of additional details of the affordable housing program information data elements.

Detailed Description of the Preferred Embodiments

Fig. 1 shows the system architecture of the present invention. A data provider or user interfaces with the system through a workstation 1 using browser software. Workstation 1 is connected through a wide area network (WAN) 2 to a communications server 3. Communications server 3 functions as a channel service unit/data service unit that is connected to the WAN 2 or other computer driven network, such as the Internet. Communications server 3 may be, for example, an Ascend MAX4004 computer.

As also shown in Fig. 1, a firewall server 4 is connected to communications server 3 through a communications server LAN 5. A WEB and FTP (file transfer protocol) server 6 is connected to firewall server 4 through a WEB and FTP server LAN 7. Database server 8 is connected to firewall server 3 through a database server LAN 9. Also connected to database server 9 are development workstation 10 and development and staging server 11. A printer 12 is shown connected to development workstation 10 and a tape drive 13 is shown connected to development and staging server 11.

The network architecture shown in Fig. 1 is not intended to show each element of the system in detail, but rather to convey an overview of the system architecture. In particular, the communications server 3 and firewall server 4 could be combined. Also, it is intended that the system be managed by an administrator staff person, hereinafter referred to as a data or system administrator or administrator staff person. The workstation for the administrator could be either the development workstation 10, staging server 11 or anther workstation not shown in Fig. 1. The administrator can have access to the system through database server LAN 9, which is a

secure LAN, through a T-1 line connection. Alternatively, the administrator can have access through a direct Internet IP address through the firewall server 4. Communications server LAN 5 may be an unsecure LAN. Further, WEB and FTP server LAN 7 is preferably a secure LAN.

Part of the security provided by the system of the present invention resides in the separation of WEB and FTP server LAN 7 and database server LAN 9, and more particularly the requirement that data exchanged between LANs 7 and 9 pass through firewall server 4. In this way, the WEB and FTP server 6, which is accessible to the WAN 2, is not connected to database server 8, which stores the proprietary asset data. Reports that are generated in response to queries from a user are output from database server 8 to WEB and FTP server 6 for access by the user. Firewall server 4 prevents direct access by users coming into the system through communications server 3, to database server 8.

According to the present invention, users, such as the general public, are able to access a WEB page via HTTP that is maintained on the WEB and FTP server 6. That is, the firewall server 4 allows communications by query users to the WEB and FTP server 6 for requesting and receiving reports. Accordingly, a user name/password combination is required at the WEB server level. On the other hand, it is anticipated that administrator staff and data providers will have access for FTP sever functions and therefore will require user name/passwords at the firewall level in order to prevent unauthorized access to the services performed by this part of the system, i.e. access to the database server 8 or staging server 11.

Access to the database server 8 by the WEB and FTP server 6 is preferably through Oracle TNS and SQL net. The database server 8 preferably will run an operating system such as Microsoft's Windows NT operating system and the database management system by Oracle, for example. Netscape is a

suitable WEB server and each of the server machines is of a type suitable for the application in view of frequency of use and performance required. The hardware of the system is easily upgradable in view of the architecture shown in Fig. 1 and a system monitor maintained at a data administrator workstation monitors and warns the system administrator when various components of the system are over utilized, when excessive paging occurs or when disk space is low. Further, each of the servers is of the type having multiple disk drives and running RAID type fault tolerant operations.

Firewall server 4 preferably runs on Check Point's Firewall-1 software on a Microsoft Windows NT operating system. Plural network cards are used in this machine to provide security for the Oracle database run on database server 8. Preferably, one network card will be connected to the communications server LAN 5, another to the WEB and FTP server LAN 7 and yet another to the database server LAN 9.

Although any suitable WEB server software can be used to host the system WEB site on server 6, preferably the WEB server 6 runs Microsoft Windows NT operating system and Netscape Enterprise server with live wire for the WEB services. Preferably also, the FTP service is run by Microsoft's Internet information server.

According to the architecture of the system shown in Fig. 1, the firewall server 4, WEB server 6 and database server 8, as well as communications server 3, can be maintained in a data center separately from or together with the development workstation 10 and development and staging server 11. A data administrator can gain access to the system through communications server 3, for example by the Internet or through a dedicated line to the communications server. Also, the data administrator can perform work on the workstation 10 and server 11 through direct connection to database server LAN 9, which preferably is not directly connected to the Internet for enhanced security. Further, although the staging server

11 is shown separated from the database server 8, its function is to store data that is to be processed before being entered into the database. Accordingly, the staging server function can be incorporated in another server, including the databse sever 8.

Although the architecture of the system shown in Fig. 1 includes separate LANs 7 and 9, this is the preferred arrangement for security purposes. However, the database server 8 and WEB server 6 could be connected directly to the firewall server 4 or connected to it through a single LAN.

Fig. 2 shows an overview of the data flow in the system shown in Fig. 1. Database server 8 of Fig. 1 is shown as a database 20 and a data warehouse 21, which are separate from each other. Database 20 is preferable an Oracle database and data warehouse 21 is preferably an OLAP MDBMS data warehouse.

When a user accesses the system through WAN 2 to query the database, either user-defined reports or pre-aggregated reports can be requested. Generally, pre-aggregated reports are stored in data warehouse 21, whereas user-defined reports are generated by processing of data stored in database 20.

Data is entered into the database system from a data administrator workstation 28 or by a data provider through WAN 2 as data files 23 in a data load process described in greater detail hereinafter. The system includes many workstations 1, although only one is shown. It is understood that an example of such a workstation is a personal or business computer supporting browser software and having Internet communications ability.

First, the data files are loaded into WEB server 6 as stored data 24 that is then loaded into database 20 using SQL loader software. The loading of stored data 24 from the WEB server 6 to the staging server 11 or database 20 of database server 8 is through firewall server 4, as shown in Fig. 1. Once data is stored in the staging server or database, it is subject to several processes, such as a validation of data

process 25, an asset matching process 26 and a reporting process 27 that determines whether or not a requested report can be made available without violating the confidentiality rules. When the validation of data process 25 determines that an error is present in the data, the stored data having the error is forwarded to the data administrator workstation 28, and if correctable, is stored in database 20 as corrected data at a later time. If the data cannot be corrected, then the data administrator informs the data provider accordingly.

The data administrator hosts a WEB page on the WEB server 6 that is accessible to the public through the WAN 2, such as the Internet. Users of the system enter into the system for the purpose of providing data or for receiving a report according to a query process.

Figs. 3-7 show a user interface for making a search query of the data stored in databases 20 and 21. The user interface is supported by browser software stored on the data provider/user workstation 1. First, a user name/password page is presented to the user before the multitab page shown in Figs. 3-7 is accessed.

In the pages shown in Figs. 3-7, a query interface is presented from which custom queries can be requested. Custom queries are produced using an interactive structured query language (SQL) tool, however different tools may be used. Fig. 3 shows the WEB page enabling geographic area or market data to be entered. Fig. 4 shows the WEB page enabling property information to be entered. Fig. 5 shows a page enabling a time period for the report to be specified and Fig. 6 shows a page enabling financing information to be specified.

With the reports available from the system of the invention, information can be made available regarding risk, returns, and best practices in financing and managing real estate, for example in the multifamily housing industry. The system provides information, using computer driven network

access, on the performance of multifamily housing assets (properties and loans) that balances the confidentiality requirements of participating firms with the ability to perform broad and useful analyses. The availability of this information in an environment in which the confidentiality risk of providing the information is minimized will satisfy a need not yet adequately met in the multifamily housing business.

Preferably, the database will contain data provided by members of the real estate industry, such as the multifamily housing industry, including property owners, property managers, lenders, and loan servicers. The following detailed description of the invention is related to the multifamily housing industry, but the invention is not limited thereto. Subscribers or members of the system are assigned a log-in identifier that can be used to gain access to the system or administrator member services. The system will also support the assignment of an ID to non-members that will allow them to generate reports. It is expected that non-members will generally not be data providers. The data providers are responsible for submitting data to the system and therefore receive a unique log-in identifier that allows them to access read and write subdirectories used for data transfer. In the event that a data provider is a large organization, company or firm, individuals within the data provider organization will have IDs providing different levels of access to the system. The system will be available to provide both on-line and custom reports that can be requested through the user query interface. Preferably, standard on-line reports generated by the system provide summary and detail income and expense data for a selected market, respectively, as shown in Figs. 50 and 51. The reports also contain basic property information,

including average number of units, vacancy rates, and effective market rents.

As opposed to custom query reports, pre-aggregated data requests are typically special tabulations from the computer database of the system that create electronic data files summarizing performance data. Pre-aggregated data requests allow a user to select static data that is grouped together in some fashion and joined with non-static data belonging to that group.

Example: Create a summary income and expense report (non-static data) for assets in the Northeast United States (static data). Group the assets by building type (garden, high-rise/mid-rise, townhouse).

Pre-aggregated data sets and reports are prepared from the data stored in database 20, and then stored in the data warehouse 21. Accordingly, the pre-aggregated reports are accessible in response to a query from a user.

The system also supports batch queries that are requested by a user. In response to such a request, a set of standard reports for predetermined queries at regular time intervals (quarterly, semi-annually, annually) are provided.

Example: Create detail income and expense reports each quarter for each of the regions in the United States where properties have less than 150 units and are not FHA insured. Create a second set of detail income and expense reports for each of the regions in the United States where properties have 150 units or more and are not FHA insured.

Requests for batch query processing are preferably automatically executed off-line at the time intervals specified. Script files define the selection criteria for batch reports.

Confidentiality Rules

Data providers supply highly confidential data to the system predicated on assurances that their data remains secure. Security features that are typical in the art are employed to ensure such confidentiality. Further, confidentiality rules, which control whether a report is output or not, based on the underlying asset information on which the report is based, are also used in generating the reports. These confidentiality rules are intended to prevent disclosure of asset level information in reports, or the discovery of such information by combining and decomposing statistics obtained through reports.

An example of one such confidentiality rule is the "5/3" confidentiality rule. In general, query results and reports from the system present only aggregate information. Asset level data can be determined from aggregate data if only one underlying asset is used to generate the report. Also, certain underlying asset information can be determined by deduction from an aggregate report if a requester is a data provider of one asset and the report is based on only a limited number of assets, particularly if there are only two assets, one of which is an asset with which the query requester has familiarity.

To prevent the disclosure or discovery of asset level data, therefore, aggregate results are returned only upon satisfaction of a confidentiality rule. The confidentiality rule is set by the system and has two conditions to be satisfied, mainly the data of the report must satisfy the following conditions: (1) The data must be based on a predetermined number of assets; and (2) The data must be based on a predetermined number of independent data providers.

For example, preferably an aggregate report is provided or output from the system to the query requester, only if the

number of assets that support the data of the report are at least 5; and the number of independent data providers that are providing the data are at least 3. The number of assets and the number of independent data providers that are used in the confidentiality rule can be set according to agreement among the data providers. As a minimum, the number of assets and the number of independent data providers supplying the data should be at least three. This prevents a query from being designed for generating a report that provides data from which the underlying asset data can be determined. Further, because data on an asset may be supplied by more than one data provider, the basic confidentiality rule is refined to account for special conditions that can arise.

In performing the counting of the number of independent data providers, the "owner" of the data, not the data provider, is used as the indicator for making the determination. The owner is the provider determined to be "closest" to the data being supplied. A hierarchy for determining the owner of the data is set in a table, maintained by the system management computer. Further, both property information and loan information must separately meet the confidentiality rule, so the 5/3 rule example becomes a 5/3/3 rule, i.e. three independent data providers for the property information of the asset and three for the associated loan information. The following is an example that illustrates an application by the system of the confidentiality rule.

Example: A number of property owners supply data on their properties and their corresponding loans. However, Lender 1 also provides information on all of these assets. In this case, since Lender 1 is hierarchically determined to be the closest data provider to the loan information because Lender 1

"owns" all of the loan information, there is only a single data provider for the information about the loans for each asset, and the rule requiring that the second number of independent data providers be met for the loan information is not satisfied, i.e. the second "3" in the 5/3/3 rule is not met.

There are additional considerations that are met in implementing the confidentiality rule according to a preferred embodiment of the present invention. For example, results covering different points of time must meet the rule for all points of time that are reported. That is, calculations showing results from two or more different years require that the 5/3 rule be met in both the start year and the end year. Further, the system requires that each value in a report must be based on at least three data points (observations). If an element in a returned report is not based on at least three non-null values in the result set, this value is suppressed in the report.

An important actor in the multifamily housing industry is the Federal Housing Administration (FHA)/HUD, an insurer of loans and provider of subsidies. Much of the information about HUD's portfolio is in the public domain or is otherwise widely available. This affects confidentiality implementation in two important ways. First, any queries that request only HUD properties need only be based on a minimum of 5 assets and do not have to meet the test of three data providers. Second, for all other queries, HUD assets cannot be counted (because its data may be publicly available) toward the satisfaction of the requirement that there be three independent data providers (i.e., it becomes a 5/4 rule).

General Description of Data Content

Preferably, the system database 20 stores asset information in records made up of approximately 225 data elements related to properties, loans, and affordable housing information. Fig. 8 shows a table 50 of the basic data elements stored in the database. Some of this information is "static" and remains relatively constant over time, while other information is termed "dynamic" because the data values change with periodic updates to the database.

Static Data

Static data elements describe the permanent attributes of an asset, and changes in them usually indicate fundamental changes in the character of the asset. For example, the number of residential buildings on a property is a static data element and a change in this number would only be expected as a result of a substantial construction effort (except for corrections to errors in the original data). Such a change would clearly indicate that the basic character of a property had changed. Substantial changes to the character of an asset, such as a change in the number of buildings, means that information about the asset after the change is not comparable with information before the change. To handle this situation, an existing asset that undergoes substantial change is recorded in the database as being terminated. Because the asset derived from the original continues to exist, a corresponding new asset is created in the database, incorporating the new features of the asset with those that remain unchanged.

Dynamic Data

Dynamic data elements describe the attributes of an asset that are expected to change frequently; these will be updated through quarterly data submissions. For example, dynamic data includes operating information such as utility expenses and

rental income for properties, the amount of the unpaid principal balance for loans, and the current interest rate on adjustable rate loans.

Within the system database, these data elements form a continuous history describing properties and loans over the period for which an asset is active. When an asset becomes inactive, its dynamic information will no longer be updated, though the historical information will be retained in the database so that it is available for the analysis of trends. Also, it is expected that some data providers may provide data whose values have been annualized. The data providers will provide an as-of date and the number of months covered by the data to determine the time period covered by the annualized data.

As shown in Fig. 8, table 50 of data elements includes control file information 51. This information includes header information, such as a data provider's unique text identifier, which is assigned by the system administrator; the effective month for the loan data contained in the file; the effective year for the loan data contained in the file; and the number of records contained in the file, for example. Further, control file information 51 can include text message for the data administrator along with the name and telephone number of the person who compiled the file being submitted, all as shown in detail in Fig. 53.

Loan static information 52 is also shown in Fig. 8. Details of the loan static information are shown in tables 52a and 52b in Figs. 9 and 10, respectively.

The load dynamic information 53 shown in Fig. 8 also includes the following data elements: loan adjustable rate information 54, load prepayment information 55, foreclosure or workout information 56, all of which are shown in detail in table 53a shown in Fig. 11.

Fig. 8 also shows, property static information 57, property dynamic performance information 58, property valuation information 59, low-income housing tax credit information 60, target rent information 61 and affordable housing program information 62. The data elements included for each asset stored in the database are intended to comprise a comprehensive description of the asset. Additional data elements can be provided for each asset or some of the data elements as listed could be reduced, depending upon a specific design of the system implementation.

Figs. 12-14 show tables 57a-c, respectively, which together comprise the property static information.

Figs. 15 and 16 show tables 58a and 58b collectively showing the property dynamic performance information 58 indicated as being one of the data elements, according to Fig. 8. Figs. 17 and 18 show the property valuation information 59 in table 59a and the low-income housing tax credit information 60, target rent information 61 and affordable housing program information 62 in table 62a.

The Data Elements Descriptions for the data elements contained in system database as partially shown in Figs. 9-18, include the acceptable predefined options for fields that are multiple choice. The Data Elements Descriptions also contain the format verification and validation rules, if any, being used for the data element and the start position, field size, and formatting structure for each element.

Figs. 53-64c show the details of the control file or header (Fig. 53) and also each of the 255 data elements that are categorized and grouped together under the appropriate headings as shown in Fig. 8. For each data element, as shown, the field name, field description, validation rule, field width and sample field format are shown. The data stored for each asset is a combination of all of the data elements shown in Figs. 54-64c plus the control file shown in Fig. 53. It is understood that a data provider works with a predefined data

base to enter the data into each of the fields of the data elements by prompting, using a web page format for example. The data elements are sequential, so the starting point for each data element can be determined by adding the fixed field widths together.

Although the foregoing explanation of the data to be maintained in the computer database of the system has been focussed on financial information for real estate properties, additonal data can also be maintained. Supplemental data in this regard includes the affordable housing information shown in Table 62a in Fig. 18, participation in FHA and non-FHA-programs, and project-based tenant rent subsidies, the number of units under contract and the levels of contract rent by unit bedroom count.

The system database also maintains various supplemental data tables to support certain query requirements and administrative functions. Preferably, data tables are maintained for counties and metropolitan areas that contain geographic, population, and descriptive economic information related to these areas. User control and administrative information is also maintained. This information is used to grant access privileges and to maintain the information needed for auditing user activities and for performing various administrative functions, such as generating bills for use of the system. This information can be maintained by the data administrator and does not need to be supplied by data providers.

It is possible that several data providers may provide data on the same asset. For example, a property owner and a lender may both provide data on a property and the loans that it secures. A hierarchy of data providers is used to select data elements that are provided by the source "closest" to the data. A hierarchy for determining the owner of the data is set

and maintained by the system management computer in the manner shown by the following example.

For property data, an owner is considered closer to the data than a lender; a manager is considered closer to the data than an owner. For loan data, a lender is considered closer to the data than an owner; a servicer is considered closer to the data than a lender.

As a result of this data selection process, it is expected that most assets in the database will be a composite of data elements from different providers. This will also provide a way to "fill-in" data that has not been supplied by the data provider closest to the asset.

To determine if data of an asset that is already stored is being provided, the data of the asset being submitted is matched with the asset data already stored in the database 20. The logic for asset matching is achieved by following a succession of rules, to be explained in greater detail hereinafter. Generally, fields are compared and assets joined on the basis of matching fields. The main fields to be used for this process are zip code, city, state, street address, number of units, number of stories, and FHA number. An asset match is determined by comparing the number of matches in the fields of the assets to a threshold number. If a number of matches occurs, but it is below the threshold number, the determination must be made manually.

The system of the present invention permits users and the data administrator to perform the following processes. Administrative Processes.

An online application that allows data administrator personnel to maintain validation and asset matching rules, monitor and execute certain data load processes, and maintain user information.

Reporting Process.

In response to template queries, a user can create search criteria for two types of reports: Trend Analysis and Geographic Comparisons.

Data Load Process

The data load process verifies and loads the property and loan data provided by the data providers. Preferably, this process is completed only periodically, for example once per quarter.

The data load process is composed of the following processes.

Receive data files. This process uses manual and automated processes to receive data files from the data providers.

Load data. This process uses Oracle's SQL*Loader product to load the submitted header and data files into the database.

Validate data. This process applies validation rules against the submitted data and saves the resulting validation errors. The validation rules may be modified automatically or manually using administration tools.

Review Errors. This process allows the manual editing, approving and rejecting of submissions and assets that fail the data validation tests.

Match assets. Preferably, this is a user-guided process that identifies multiple data submissions for the same asset from different providers. The process creates a single composite record for each matched asset.

Migrate data.

This process is preferably executed once per quarter to make validated data available to the WEB-based reporting application.

The data load process involves the step-by-step migration of data through several sets of database tables. A naming convention distinguishes the tables that are used in each sub-process. Table 70 shown in Fig. 19 shows the key tables used by the data load process.

Receive Data Files

The first step in the Data Load Process is to receive data files from the data providers. Preferably, data is collected from data providers on a quarterly basis. The data providers can submit files manually via diskette or email to the system administrator and preferably directly via FTP (i.e., file transfer protocol) to the WEB and FTP server 6. Each data provider has an "incoming" FTP directory, explained in greater detail hereinafter, where transmitted files are placed. The submitted files are provided in a pre-defined fixed-width text file format or in a pre-defined Microsoft (MS) Access database file. Files may also be compressed and encrypted.

The Data Provider's File

It is an objective of the present invention to reduce the burden of submitting data by using a file format and file submission system that increases the ease of electronic data submission. Preferably, each provider receives a predefined Microsoft Access database that runs on the data provider workstation (interface) 1 for compiling the data. The database also contains export specifications that can be used to create the preferred format for the fixed-width text file for loading the information into the database system.

Data File Layout

Specifically, the electronic file that is submitted is preferably a fixed-width text file that can be automatically processed by the system. In addition, a header file that is submitted with each data file automates processing of the data

submission. A database, such as Microsoft Access database, is preferably used by the data provider at the workstation 1 to compile the data to be submitted. The database can contain the system table for compiling the submission data. A second table will be included for the header file information that is submitted with the data file. The Access database may be transmitted to the system directly, or the data providers may use the export specification file contained within the database to export the submission data and control file into a fixed-width text file format for transmission to the system.

Header File

The standard text file format requires that each quarterly submission must include two files: one header (or control) file and one data file. All submission files are named using the following convention, for example:

Provider ID + YYYY + MM + encryption designator + file designator + submission number + extension.

With reference to Fig. 20, the provider IDs are assigned to be 12 characters or fewer. The YYYYMM should be the effective year and month of the submission. The encryption designator will be either "E" for encrypted or "U" for unencrypted. The file designator will be "H" for header, "D" for detail, or "A" for MS Access. The submission number will be a sequential counter corresponding to the number of times the data provider has submitted this file. The first file will be labeled "1." The extension will be either "txt" or "mdb" for text and MS Access submissions, respectively. A period will separate the file name and the extension. The redundancy in this naming convention is intentional. Data providers using an operating system without long file name support will use their FTP client to create the correct name on the system FTP Server.

Example: AJACKSON first runs the C program (discussed below) to make sure the submission is in the correct format. Then, he optionally encrypts the file. The data provider ID for John Smith Management Company is "SMITHMGMT." Assuming he has a pair of unencrypted header and detail files for his first submission effective 12/99, he will name them

SMITHMGMT199912UH1.TXT

SMITHMGMT199912UD1.TXT

Ensuring Correct Format: File Verification

The FTP directories are checked for new submissions. When a new submission is detected, a copy is made of the file from the FTP server to the database server. If necessary, The system will unencrypt the file. A file verification utility program (C program) is executed that performs very basic format checks on the data.

Examples of the format checks include scanning for correct record length, the presence of carriage returns in the file, and the presence of both a header and a detail file. The C program does not perform data validation checks since this utility functions for the purpose of ensuring that the system can load the data successfully. (The program recognizes that UNIX files and PC files use different row terminators.) The system administrator distributes the C source code and Intel binaries for this program to the data providers. The data providers must run this program and correct any errors it identifies before submitting a file. This program ensures that SQL*Loader can load the data into a table of the same configuration as the input file, with all columns allowing nulls.

Data Receipt Notification

Once the file verification program runs, The system will send an email note to the point of contact (POC) listed in the

header file, and it will send an email note to the POC named in the DATA_PROVIDER table, if they are not the same. The note will detail the following:

Date and time of the submission

Status (success or failure) of the C program

Error messages from the program, if any.

Other users authorized by the data provider, with approval from administrator, may request that the system notify them when it has received a file.

File Archiving

After processing a file, The system will copy it into a subdirectory belonging to a data provider's root directory. These subdirectories will be named by quarter, for example, 99QTR4. They will be read-only for the data provider. The system will rename the files as it copies them to preserve version information, such as HEADER1.DAT, DETAIL1.DAT, etc. By looking at these directories, providers and administrator staff can see a complete history of previously submitted files. All data will be loaded via SQL*Loader into the SUBMISSION and INPUT_FILE tables in the database.

Before processing a file, the system will copy files to the staging database server. The database server will have a node with directories named for the data providers, and subdirectories named by effective year and month, for example, 199912. An administrator staff member will be able to read these files, but data providers will have no access to them. To the extent allowed by disk space, these files are always present on the database server. If storage constraints demand that old files be removed, they can be archived on digital linear tapes or other media.

More specifically, submission files that have been placed in the submission directories are backed up. Submission files are moved from the provider's incoming ftp directory in the

WEB and FTP server 6 to the "_fakeftpdir" directory. The files are also copied to the provider's backup directory "Backup_ftp" and then removed from their original location (to prevent reloading). In order to receive data from a new data provider, the copying.bat file needs to be updated manually.

Fig. 21 shows a data flow diagram of the data submission process. As shown, data of data providers 80 is written to write directories 82 of the FTP server 6 after being optionally un-encrypted by encryption module 81. This presumes, however, that the data 80 is in the standard text file format for FTP transmission. In this case, the FTP files can be written directly to server 6 using the data provider/user workstation 1. On the other hand, if the data is not in the standard format, then it is submitted to the data administrator 83 by hand or e-mail, for example, which is then processed into standard format and written into the FTP server write directories 82. Upon completion of the writing of the data 80 to the FTP server write directories 82, a submission log entry is made in submission log 84 and the data is copied into pre-staging development database server 85, which can be part of the development and staging server 11 shown in Fig. 1 or part of the database server 8. At this time, if the data has been encrypted in encryption module 81, it is un-encrypted in un-encryption module 86 and moved to the SQL loader 87. The call flow and tables used during this data load process are shown in Fig. 22.

As explained with reference to the Procedures and Tables shown in Fig. 22, a batch process called loading.bat is launched periodically, every hour for example, to load and validate the files that were submitted in the previous hour. The loading and validation processes are performed for one data submission file at a time until all of the files submitted in the previous hour are processed. The flow of the loading.bat process is shown in Fig. 22 beginning with the

execution of the load_data.exe command which sets forth the execution of the steps of pgld_loadata_pkg through prld_ins_aptdata_input.

Specifically, the loading process uses the Oracle SQL*Loader utility to insert all of the records in a submission data file into the ld_1submission table in the Oracle database. This table holds the records for one submission data file at a time. The table is emptied before data for each new submission is loaded.

The SQL*Loader process is controlled by a file called APTDATA.ctl (not shown). This file contains the specifications that match each field in the input file with the corresponding column in the database table.

During the loading process, blank fields in the submission data file are changed to a NULL value, and date fields in "YYYYMMDD" string format are converted to Oracle's DATE format. The load process records load statistics in a file that has the same name as the input data file with a ".log" extension. Errors are recorded in files that have the same name as the input data file with ".bad" and ".dis" extensions.

Once the data file for a submission is loaded, header file information is inserted into the ld_submission table. A unique, sequential submission_id is generated and used to identify each data submission in the table. The status_flag column in the table is set to the value 'L'. The provider_name field in the table is set equal to the company_name field in the WEB_company table for the same provider_id as the one provided in the submission header file.

The last step in the data load process, is to copy the submission data from the ld_1submission table to the ld_APTDATA_input table. At this stage, each record in the

latter table is assigned a unique, sequential record_id. The new submission data is appended to the ld_APTDATA_input table.

Data provided by the Department of Housing and Urban Development (HUD) is preferably processed somewhat differently. HUD provides an additional file, called the HUDOnly file, which contains financial attributes provided only by HUD. A manual procedure is required to load this file.

Data Provider Accounts

A separate FTP account is established for each individual of an organization who is authorized to submit data for a data provider. On the WEB server 6, there are two separate directories for each data provider. There is a write-only directory 82 (Fig. 21) for submitting data and a read-only directory 97 (Fig. 23) for retrieving other files, such as error reports. Security requirements limit access by each provider to their specific directories for sending and retrieving files. The data provider account is preferably configured under Windows NT with no list privileges to prevent data providers from viewing or accessing other directories.

Example: Suppose John Smith Management Company authorizes a user named Andrew Jackson, with a user ID of AJACKSON, to submit and retrieve the company's data. When AJACKSON connects to the WEB server 6, he will be in "Smith Mgmt Write." He can set his transfer type (ASCII or binary, as appropriate) and immediately transfer the files to the write-only directory 82. If he wishes to retrieve a file, he must execute the following command:

CD "Smith Mgmt Read"

The home FTP directory for each user is the write directory 82 for the associated data provider. Directories will be named using the COMPANY_NAME column in the COMPANY table of the database, suffixed with the word "Read" or

"Write." These directories are created as subdirectories under the \InetPub\FTProot\Data Provider\ directory. For each directory, a virtual directory will be created and named COMPANY_NAME + suffix. Accordingly, on the WEB server 6, user Andrew would find two directories:

\InetPub\FTProot\Data Provider\Smith Mgmt Read\

\InetPub\FTProot\Data Provider\Smith Mgmt Write\

There would be two virtual directories for these physical directories:

Smith Mgmt Read

Smith Mgmt Write

Ensuring Data Integrity

To help ensure the integrity of the data being provided, the system will run a series of validation checks on the data. This validation procedure evaluates and refines the data.

Fig. 23 shows the data flow of the data validation process. Data from the SQL loader 87 comprises good records and bad records. The good records are stored in the pre-staging database load tables 90 from which the data is checked in the data validation process by the PL/SQL data validation module 91. Validation of the data is according to certain pre-defined rules.

After a data provider file has been loaded into the pre-staging database 90, data validation program 91 is run to generate validation report 93 for optional manual review by the system administrator 83. The data validation executed validation rules from a validation rules log 92 to make element-level checks, input-record-level checks, substantial changes to static data checks and to make checks for duplicate record detection and new record detection. Parameters for data validation are examined for their ability to detect erroneous data.

A complete list of the data validation tests that will be run against the data are shown in Figs. 25-40and further include checking whether the header file format is entered correctly. The validation check suppresses recurring validation error messages for exceptional values that have been verified as correct without limiting the information available for other values.

Data Validation Processing

After the system has successfully loaded a data provider's file into the INPUT_FILE table, the validation process continues. As SQL*Loader loads a file, Oracle may generate errors. After SQL*Loader is finished, PL/SQL procedures validate all data.

Single-Row Validation Tests

The first set of validation tests will be run on each record submitted. These tests compare the values submitted against a set of "validation rules" that identify a range of acceptable values. If the values submitted are outside the parameters identified by the validation rule, an error will be recorded. An example of the validation rule application is as follows.

Example: Number of units must be between 5 and 10,000

Elevator flag must be Y, N, U, or null

If loan adjustable rate information is populated, interest rate code must be "A" (adjustable rate)

Changes to Static Data

In addition to applying the submitted data against the set of validation rules, each record will also be compared against its previous submission. If any values for static data elements are different than the values previously submitted, an error will be recorded. Manual confirmation of each change

to static data elements with the data provider is preferred before the record is loaded into the database.

Missing Loan Test

There will be several multirow validation tests. The only one being used initially is the missing loan test, which covers closed loans. When a loan closes, it is preferred that a data provider submit data for the loan, one last time, in the first quarter after the close of the loan. This record provides a loan closeure date and loan termination code. When a provider fails to submit data for a loan without providing such a closeout record, a loan is "missing." The system will record all missing loans as errors for review. Conversely, when a loan that was closed is resubmitted as being active, an error is generated. Continuing to submit the same closeout record for multiple quarters after a loan is closed is not an error.

Validation Reports for Data Providers

The system will email a summary of errors to the data provider and system administrator. This summary will give only a text description of the validation rule and the number of records that failed to meet the test. This summary information will not violate the confidentially of the data.

This validation information will also be available via the WEB. Since the WEB server uses HTTPS to encrypt information, the data providers will be able to see the detail (individual rows) for each error. Data providers will have to log on to the WEB server to see errors and will be limited to viewing only errors for the data they have submitted. The system administrator staff will have access to all data exceptions reports for all data providers.

Validation Rules

Data validation is performed as part of loading.bat immediately after a submission file is loaded into the database and before the next submission file is processed. Data validations are performed by executing a series of predefined SQL statements that represent validation rules. For each statement, the application logs the number of records that fail to meet each validation criteria.

Before executing the validation rules, every asset data record for the current submission_id is copied into the val_APTDATA_input table. Validation rules are executed against the data in this table. The following validation rules are performed in sequence:

Domain rules (validate_chardomain). Ensure that a string value in a field is a valid value for the field. For example, domain rules check that the workout_facility field has either a 'Y', 'N', or 'U'.

Formula rules (validate_formula). Ensure that calculated field values are correct. For example, formula rules check that the debt_service field equals interest_paid_amt plus principal_paid_amt.

If-then test rules (validate_itthentest). Ensure that if data for one field meets a condition, then another related field meets another condition. For example, if no_stories > 6, then elevator_flag = 'Y'.

Range rules (validate_range). Ensure that a field value falls within a specified value range. For example, initial_construction_yr is between 1850 and the current year.

Foreign key rules (validate_foreignkey). Ensure that a field value matches a list of valid values in a lookup table. For example, foreign key rules make sure that the submitted zip code matches a valid zip code in the zip table.

Custom rules (validate_custom). Enforce custom rules that do not fit in any other category.

The number of records in a submission that do not pass a particular validation rule are counted and recorded in the val_validation_error table. A record is inserted in this table for each erroneous value found in the tested field. The error record contains a count of the number of records in the data submission that contain the erroneous value.

For example, if there are five records in a submission that contain the value 'W' in the workout_facility field, and two records that contain the value 'G', then the val_validation_error table will have two error records, one with a count of five for the 'W' value and one with a count of two for the 'G' value. The error record also contains the submission_id, a code identifying the type of test performed (val_type) and a number identifying the validation rule (val_number).

Once the validation rules are performed, static data in the submission is tested to see if the submitted values are different from the values previously stored in the database. Static data are asset attributes that do not normally change from submission to submission (for example, a property's address). Two additional procedures are executed to verify static data:

Static property rules (validate_static_property). Determines whether any static property information has changed from the last submission.

Static loan rules (validate_static_loan). Determines whether any static loan information has changed from the last submission.

The static data values for each asset in a submission are compared to the last accepted values for that asset, which are

stored in the "before matching" tables: bm_property and bm_loan. Any differences are logged in the val_static_error table. For each changed static data column, this table contains the current value copied from the corresponding "before match" table and the value copied from the val_APTDATA_input table.

The validation process works by executing several predefined stored procedures. These stored procedures contain static SQL statements that execute the individual validation rules and log the results. The validation rules can be changed using an administrative interface, in which case the validation stored procedures need to be regenerated.

Data Cleansing

The data administrator reviews data exceptions reports from submissions that do not pass validation tests. From this, the scope of errors and the possibility that systematic problems exist within the process of data file creation are determined. The system administrator staff notifies data providers of errors, and provides a complete errors report. This error report is also available on-line in the data provider's read-only directory. After review of the report, the data provider and administrator determine the best way to correct any data discrepancies.

Review Errors

After the validation rules are executed against one or more submissions, the data validation errors can be reviewed manually by the administrator staff using a browser interface. The browser interface allows clean up of data records so that they can be accepted and moved on to the asset matching process. During error review, the administrator can edit data directly to correct errors, choose to accept records even if

they fail one or more validation checks, or reject records that have failed validation checks.

Figs. 25-37 show the validation rules that are performed in the SQL statements set forth in Fig. 24. The validation rules are performed on load static information as set forth in tables 250a and 250b in Figs. 25 and 26. The validation rules for loan dynamic information are set forth in table 270 as shown in Fig. 27. The load adjustable rate information validation rules are set forth in table 280 in Fig. 28 and the loan prepayment information validation rules are set forth in table 290 shown in Fig. 29. In table 300 in Fig. 30, the foreclosure or workout information validation rules are set forth.

The property static information validation rules are set forth in tables 310a, 310b and 310c, respectively shown in Figs. 31, 32 and 33. The validation rules for the property dynamic information are set forth in tables 340a, 340b and 340c, respectively shown in Figs. 34, 35 and 36.

Tables 370 and 380 describe the validation rules for the target rent information and affordable housing program information, respectively, as shown in Figs. 37 and 38. Figs. 39 and 40 show the validation rules for the property valuation information in table 390 and the validation rules for the low income housing tax credit information in table 400, respectively.

Fig. 41 shows a flowchart of the procedure of reviewing errors. First, in a step 410, administrative options are selected. These options are presented to a system administrator staff member who accesses the system as before-mentioned, preferably through the host web site.

The first HTML page used for error review is the Submission Log page (validation.html) 411, which displays all

of the current submissions in the system. The page combines all of the current submissions in the ld_submission table with the total number of validation errors recorded in the val_validation_error table. The combination of these two tables gives the data administrator a summary view of the potential number of errors in the database on a per submission basis. For each submission, the data administrator can see on the screen the provider information and the total number of validation errors found for the submission.

In order to learn more details about the particular errors in a submission, each submission shown in the table has a link to the Error Log page (validation_result.html) 412. If a submission has not yet been accepted or rejected, then the Error Log will display a summary of all validation rules that failed for this particular submission. All of the validation errors recorded in the val_validation_error table are displayed to the user, along with the rule itself. On the Error Log page 413, the data administrator can accept or reject the entire submission (which updates the status_flag for the submission in the ld_submission table). Alternatively, the data administrator can click on a rule to view the particular records that failed the validation test.

If the data administrator tries to use the Error Log page to view a submission that has already been accepted, then this page simply confirms that the submission has been accepted. The administrator cannot change the status of the submission once it is accepted or rejected.

When the data administrator clicks on a validation rule, the application opens the Detail Error Log page (validation_detail.html) 413. The Detail Error Log page displays all asset data records in a particular submission that failed the validation test. The page allows the data

administrator to accept or reject individual records in the submission, suspend a validation check for a submission, or reverse a suspended validation check.

When the Detail Error Log page is loaded, the val_APTDATA_input table is first searched to find all records that do not pass the current validation check. In order to show only those records that have not been marked as accepted already, the bm_property table is used to filter out previously accepted records. The val_error_detail table may also be used to identify records in the current submission that have been previously marked as rejected. The data administrator can choose not to view these records.

The data administrator can accept or reject one or more records on the Detail Error Log page by selecting a radio button next to each property and clicking on A Save Changes button, for example. When this happens, a record is inserted into the val_error_detail table to record the appropriate change to the accept_or_reject_flag ('A' or 'R'). Each record shown on this page contains a link to the Insert Table page (updateData2.html) 414. The Insert Table page allows the data administrator to edit the fields in a submission record directly. Any such editing results in an entry in the Summary Error Log (validation_summary.html) 415,

The result of the validation checking may also result in static changes being made by record 416) or static hanges made by column (417), after which the entriy is entered in a validation static column detail page 418.

Major Revisions

In the case of overwhelming data errors or numerous systematic problems, the data provider is notified to recreate the data file and resubmit the data. In this instance, the validation process would start over.

Minor Revisions

The system is intended to be user friendly for data providers. If smaller revisions to the data are needed, the data provider and the administrator staff member may agree that the administrator staff member will manually adjust the data. The data provider then makes any changes necessary to the source data for future submission, and the administrator staff member keeps a detailed log of all conversations with data providers to verify, update, or correct information in the database, including a record of the erroneous data and the corrected data. Administrator staff will manually adjust data through a WEB interface that will track all changes made to the data and provide a log of who made the changes and when. Written confirmation of these changes will be forwarded to the data provider. Changes are made to data elements in the system with the verification and acceptance of the data provider. If the data provider is unable to resolve the issue, administrator may determine the data is insufficient to include in the database. The deletion of this record would also be included in the detailed record of data changes.

Asset Matching

Fig. 42 shows the data flow and performance of the steps followed in the asset matching process. As part of the asset matching process, the data records, which have already been subjected to validation checking, are stored in the development database 11. The system administrator triggers asset matching in step 420, which results in the asset matching being performed in PL/SQL data matching step 421. After the asset matching is performed, the data is again stored in the staging data base 11. The system administrator checks the data stored in staging database 11 in step 422 for reviewing and refining the asset matching that has been

performed. Following this step, the asset matching is performed again by returning to step 420, or if needed, otherwise the asset data is cleansed by the system administrator in step 423 and the flow passes to a step of applying the data to query tables in step 424. Then, the data is stored in the production database 20 as validated, and asset matched data. From this data, PL/SQL batch queries following the confidentiality rules are processed in step 425 and reviewed by the system administrator and also PL/SQL data administrator indices are determined in step 426 followed by review of the system administrator. After these indices and queries steps are completed, pre-aggregated reports are stored in data warehouse 21.

Also, data stored in the production database 20 is processed in step 427 using PL/SQL to make pre-aggregated data sets in which the confidentiality rules enforced (step 427). The system administrator reviews the data sets in step 428 and the pre-aggregated data sets 429 are written to the data set subscribers write directories 82 in the FTP server 6 so that data set subscribers can retrieve the pre-aggregated data sets through FTP. Alternatively, the pre-aggregated data sets 429 are made available over the WEB 2 to the data set subscribers, which also receive an e-mail notification.

In detail, the asset matching process is composed of three distinct sub-processes: Before Matching, Matching, and After Matching.

The purpose of the Before Matching process is to merge newly submitted property and loan data with pre-existing property and loan data. Before this stage, a single record contained all of the data fields for the asset. The Before Matching process splits the single asset record into four

separate records: property static, property dynamic, loan static and loan dynamic data.

With reference to Fig. 43, which shows the Procedures and Tables and for each new data submission, a stored procedure called prbm_update is launched automatically or manually by the system administrator to perform this merge, which includes the steps shown from prbm_update_property through prbm_update_loan_dynamic, followed by prbm_calculated_columns. If there are any new properties in the submission, then their static fields are inserted into the bm_property table. The val_static_error table is then searched to find any updates to existing static property information. If the data administrator had reviewed and accepted a change to a static field during the Review Errors process, then an update will be made to the existing record in the bm_property table.

After the static property data has been updated, all of the dynamic property data records for the submission that have not been rejected are copied from val_APTDATA_input to bm_property_dynamic.

After the property data has been merged into the before matching tables, loan data is merged in a similar manner. If there are any new loans in the submission, then their static information is inserted into the bm_loan table. The val_static_error table is then searched to find any updates to existing static loan information. If the data administrator has reviewed and accepted a change to a static field during the Review Errors process, then an update will be made to the existing record in the bm_loan table.

After the static loan data has been updated, all of the dynamic loan data that was not rejected is copied from val_APTDATA_input to bm_loan_dynamic.

A separate stored procedure, prbm_calculated_columns, must then be manually executed to complete the Before Matching process. This procedure computes certain calculated values (for example, total revenue) for each asset and updates these values in the bm_property_dynamic table.

Matching

The Matching process involves running several procedures that identify data submissions for the same asset from different providers. Each procedure, called an iteration, uses one or more matching criteria to determine the likelihood that a group of two or more records are really for the same property. For example, an iteration may check whether the property names and addresses for two or more asset records are similar. The data administrator can accept or reject a proposed set of asset matches via a WEB browser.

As shown in Fig. 44, to start the matching process, a perl script called before_matching.sql is executed manually or automatically by the system administrator. This perl script uses the tables ma_match_property through ma_final_match_sets in connection with executing a group of stored procedures (ma_before.pl) that perform the asset comparisons. Three stored procedures (prma_match_iter1, prma_match_iter2, and prma_match_iter3) execute different match criteria against the bm_property table. Assets that meet the matching criteria in these three iterations are grouped together as possible asset matches.

For each possible match combination found, a record is inserted into the ma_match_property table. The prma_assign_match_key procedure, which runs after the three iterations above, assigns every potential match combination a match_key number, which groups the matching records.

After each potential match combination is identified, the prma_working_matches procedure inserts a single record into the ma_working_match_sets table for every proposed match group. The mifhi_property_id for each property in a proposed matching group is associated with the match_key in the ma_working_match_sets table.

The matching procedures contain static SQL statements to perform the comparisons. These procedures would need to be regenerated when the data administrator changes the iteration rules in the administrative interface.

Once the matching procedures are executed, the data administrator uses an administrative interface (at the web site) to accept or reject the proposed matching groups.

At the web site, a first HTML page that is can be used for asset matching is a Search Proposed Matches page (asset_matching_search.html). The data administrator can select which column he or she wishes to use to filter the set of proposed matches. Typically, each of the three iteration numbers is used as a filter. The Proposed Matches page (proposed_matches.html) then displays summary information for each set of properties that have been identified as potential matches. The data administrator can accept or reject one or more matched sets of assets on the Proposed Matches page by selecting the appropriate radio button next to each set of proposed matches and clicking on a Save Changes button, for example. When a proposed match is accepted or rejected, the corresponding accept_or_reject_flag ('A' or 'R') is updated in the ma_working_matches table. After all of the proposed matches have been reviewed and rejected or accepted, the administrator can run the After Matching process.

During this process, the ma_final_match_sets table is checked to make sure that proposed matches that had been

previously accepted or rejected are not presented to the user again.

After Matching

The objective of the After Matching process is to combine data for assets that were accepted as matching assets by the data administrator. For each set of matched properties, the process creates a composite asset record out of the data for the assets in the group.

The After Matching process is executed automatically or manually by the system administrator and will be explained by referring to Fig. 45. The first script manually executed during the After Matching process is after_matching.pl, which refers to tables ma_match_property through ma_working_working matches, as shown. This script calls ma_after.sql and matching.sql to add the matched assets in the ma_working_matches and ma_working_match_sets tables to the ma_final_matches and ma_final_match_sets tables. During this process, sets of assets that were accepted or rejected during the Matching process are inserted into the ma_final_match_sets table so that in the future these match sets will be automatically accepted or rejected.

The second script that must be executed is migration1.pl, which refers to tables temp_WEB_property_dynamic through bm_property, as shown in Fig. 45. This script calls the pram_update_property and pram_update_property_dynamic stored procedures as part of the call to am_properties.sql. If an asset record was not matched with any other records during the matching process, then these scripts will copy records directly from the bm_property and bm_property_dynamic tables into the am_property and am_property_dynamic tables. For each of the accepted asset matched records, these procedures will select the minimum values from each of the columns in the

accepted matched records to create a composite record. The composite records built from these matched sets are inserted into the am_property and am_property_dynamic tables.

The migration1.pl script is used to remove duplicate records from the am_property_dynamic table. This process creates two temporary tables, temp_p_d_distinct and temp_p_d_distinct_pk.

With reference to Fig. 46, in a subsequent migration process, all of the asset information is copied to the production database used for reporting. The migration process is executed, preferably manually, by launching the migration.pl script (not shown), that calls the stored procedure prWEB_temp_p_d to build a temporary table called temp_WEB_property_dynamic out of the data in the temp_p_d_distinct and temp_p_d_distinct_pk tables.

Once this step is complete, the script migration2.pl must also be executed, preferably manually, which calls the WEB_properties.sql procedure. This procedure first remove all records in the WEB_property and WEB_property_dynamic tables. The procedures then copy all of the records from the after matching tables into the WEB_property and WEB_property_dynamic tables. Once this process is complete, the reporting process will use the updated asset information.

Administrative Processes

Fig. 47 shows the function of the data administration performed by the system administrator. The user information of the system users and the data provider information of the data providers can be performed by the system administrator 470. For example, a staff person of the system administrator can log onto the system using an admin log in page 471 which authorizes the administration functions by passing the user to an admin home page 472 unless the log in fails producing a

suitable notification 473. Then, the system administrator can select one of the WEB pages 474, 475 or 476 for making changes to user information, data records of the assets or data provider information, respectively. The functions enabled by the system for maintaining this data are shown in Fig. 47.

Edit Validation Rules

For example, system administrator staff may use the administrative interface to edit the rules used during the validation process. There are four types of validation rules that the data administrator may edit via the administrative interface: domain rules, formula rules, if-then-else rules, and range rules. Several database tables contain the current validation rules used. There are different tables for each type of validation. See Fig. 24. The administrative interface updates the information in these tables.

The first part of the update process involves using a WEB interface, explained with reference to Fig. 47, to change or add validation rules. For each type of rule, there is a WEB page, accessible via the administration interface under Validation Parameters, for example. As a single validation rule is changed or added, the appropriate validation rule table (such as val_chardomain) is updated to reflect the new set of validation rules.

If the data administrator needs to change a foreign key or custom validation rule, then he or she would have to edit the appropriate validation tables directly.

Once the data administrator has made all of the validation rule changes, the validation_first.pl script must be executed manually to backup the old validation rule scripts. The validation_stmts.sql script must then be manually executed. The stored procedure that this script executes, prval_define_from_template, regenerates the stored

procedures for each set of validation rules (such as the stored procedure validate_chardomain). Once all of the validation stored procedures have been regenerated, the script validation_final.pl must be executed manually to load the new stored procedure into the database.

Edit Asset Matching Iterations

From another WEB page, not shown, such as an Asset Matching Parameters WEB page (asset_matching_parameters.html), a data administrator can add, edit or delete the asset matching parameters and iterations used during the Asset Matching process. The Parameters page displays all of the records in the ma_criteria2 table. The data administrator can then edit a field (such as the threshold) and save the updated field in the ma_criteria2 table. An iteration parameter can also be created or deleted from this page.

Administer Membership

This process allows users to be added/edited/removed from the system, as shown in Fig. 47. Only users that have been listed as "admin" in the user_category_id field of the WEB_person1 table may perform this function. In addition, all valid users may edit their own member information and change their password through the "Member" area of the WEB site.

Administrator staff may administer users through the WEBsite/DataEntry user interface. This interface allows the administrator to add and edit companies and persons.

The Company add and edit pages display all of the fields (such as company name, address, etc.) found in the WEB_company table. As new companies or changes are made on these pages, the new data is stored directly into WEB_company.
The Person add and edit pages display all of the fields (such as name, address, etc.) found in the WEB_person1 table. As

new people or changes are made on these pages, the new data is stored directly into WEB_person1.

Reporting Process

Fig. 48 shows a data flow diagram of the query and reporting process. In particular, a query subscriber 480 makes an inquiry to the system through the WEB site home page 481 and logs into the WEB site through a log in page 482 which either authorizes the user or notifies the user that the log in has failed at step 483. If the log in is authorized, then the user proceeds to the query templates or screens shown in Figs. 3-7 in step 484 and the result of the query is processed from the production database 20 in step 485 with TL/SQL filter. If the resulting report from the query generated in the query templates fails to pass the confidentiality rules, then the report is not forwarded to the results page 486. Reports that are forwarded to the results page 486 are printed at 487 or otherwise output from the system.

The Reporting Process component provides users with aggregated data reports based on asset information stored in the database. A user can create search criteria for two types of reports: Trend Analysis and Geographic Comparisons. The reporting application is accessible to registered users through the WEB site.

Using the report definition pages or templates shown in Figs. 3-7, the user selects the desired report criteria. The user may select the following criteria types: market, time period, property, and financing. The Search Criteria Pages procedure is executed at the beginning of a search and a stored procedure PRWEBREP is called. Tables session_query_parameter through WEB_session_query_parameter tables are referred to in the execution of the search, as shown in Fig. 49. Each search criteria selected is stored

into the session_query_parameter table. The application also uses the ESRI mapping software to present area maps to the user for selecting market areas.

Once the user has specified their search criteria, PRWEBREP is executed to generate the report. Before the report is shown to the user, the data records are analyzed to verify that the search criteria follows the confidentiality rules, such as the 5/3/3 rule (at least 5 assets provided by at least 3 data providers, and at least 3 non-NULL values per field). If the 5/3/3/ rule is met, then a report_body.html page displays the report to the user.

The main function of the PRWEBREP procedure is to call prWEB_report. The prWEB_report procedure builds the query string (based on the user's search criteria) used to search the production database for records to be used in the report. The prWEB_report_market_sql procedure executes each search criteria and stores the results in either the WEB_hold_summary or WEB_hold_detailed table (depending on the type of report requested). The procedure prWEB_compute_summary_medians or prWEB_compute_detailed_medians calculate the averages, medians, and counts for the records returned and store the results in WEB_hold_medians. Finally, the calculated results are inserted into the WEB_report and WEB_report_data tables, which are used to build the body of the report.

Each report request is identified by a session_id. From the report page, users may choose to save a report request for future execution, download a report, or run another report. If the user chooses to run another report, then they are given a new session_id so that each report requested can be tracked in the database.

Fig. 50 shows an example of a WEB page, report_body.html page, that displays a sample summary report 500 that is output

by the system. Fig. 51 shows a sample detail report 510 that is output by the system, and Fig. 52 shows a sample detail report statistics 520 that is output in conjunction with the sample detail report 510.

While preferred embodiments have been set forth with specific details, further embodiments, modifications and variations are contemplated according to the broader aspects of the present invention, all as determined by the spirit and scope of the following claims.